A Path Planning Method Based on Genetic Algorithms and Curve Fitting Techniques

Brock Kopp
Mechatronics Engineering
University of Waterloo
Waterloo, Ontario
brock.kopp@uwaterloo.ca

Karl Price
Mechatronics Engineering
University of Waterloo
Waterloo, Ontario
karldprice@gmail.com

Angelica Ruszkowski
Mechatronics Engineering
University of Waterloo
Waterloo, Ontario
aruszkow@uwaterloo.ca

***Abstract*—Navigation and path planning are critical components of developing robotic technology, where optimized path planning can improve efficiency and throughput of an industrial robotic manipulator. A genetic algorithm is used to plan a polynomial based trajectory which avoids collisions with any obstacles and minimizes trajectory path distance and dynamic constraints. The heuristic nature of the genetic algorithm allows it to determine a viable and efficient path for a manipulator quicky as it's environment changes. Algorithm results are verified by comparing with deterministic algorithms which produce the absolute best path.**

***Index Terms*—Genertic Algorithms, Curve Fitting, Path Planning, Grid Method, Robotics**

I. Introduction

Path planning is a critical component of navigation technology, a fundamental area in robotic research. Verifying path safety is critical for both the safety of the robot as well as its surrounding environment, which can contain people or valuable infrastructure. The path must ultimately reach its destination while satisfying specified criteria, the most common of which is distance (or time). Another important criteria is path dynamics, where the forces and other dynamic characteristics of the robot are considered. Excessive acceleration and jerk cause stress on the robotic manipulator as well as robot tracking characteristics which are often a function of acceleration. These criteria accounted for by relatively few common path planning algorithms [**?**].

Path planning techniques can be applied to industrial robotic manipulators to ensure efficient operation while maneuvering safely in the environment. Articulated manipulators are bounded to their configuration workspace, which is a two-dimensional representation of their allowable motion range [**?**]. Frequently used algorithms for motion planning include probabilistic roadmaps, potential field methods, and neural network approaches [**?**], [**?**], [**?**], [**?**]. Since every robot has a unique and complex task and environment, a robust solution is necessary. Genetic algorithms are especially suited for such complex optimization problems [**?**].

As a stochastic optimization technique, a genetic algorithm will find the global optimum for a problem in a reasonable amount of time and computational cost. Due to the iterative evolutionary nature of genetic algorithms, computational time may be greater than other applicable algorithms. However, the robustness of a genetic algorithm, combined with its ability to easily evaluate all path criteria of the optimization problem make it an appealing solution.

The robot's environment and obstacles will be represented in a discretized grid, and polynomial curve fitting techniques will be used to define the optimum path from the start to the end location through the specified environment. The curve fitting method helps simplify and expedite the generation of a desired path, compared to commonly used techniques such as rapidly exploring random trees [**?**]. The genetic algorithm will determine the polynomial coefficients which result in a curve with the greatest fitness, as determined by the constraints of the problem (namely path distance and jerk). A polynomial function was selected to describe the path since it inherently results in a continuous trajectory rather than traditional discontinuous waypoint based trajectories.

While there is no absolute best path planning algorithm for all situations, it is proposed that:

- A Genetic Algorithm can effectively plan a path through a robotic manipulator's configuration space.
- The algorithm presented can plan a path which not only minizes distace travelled, but also account for robotic manipulator dynamics such as acceleration and jerk.
- The algorithm can identify a viable solution which is within XXXX% of the shortest distance as determined using a deterministic algorithm such as wavefront, but in much less time.

II. Background Review

Path planning is an active area of research due to the complexity of the problem and the need for a robust solution applicable to various environments. Many algorithms exist, but none of the proposed algorithms are capable of encompassing all the problem constraints for various environments [**?**].

Existing path planning solutions include reactive motion planning algorithms, graph and probabilistic based motion planning, and optimization based planning [**?**]. Each of these has advantages and disadvantages with regards to planning a route through a robot's configuration space. Reactive planning algorithms such as potential field and wavefront methods are simple in concept however these are local approach methods

and do not consider dynamic constraints. Some are inherently susceptible to local minima, making them unacceptable for obstacle avoidance [**?**]. The wavefront method is designed to find the shortest path, and as such will be used as the benchmark against which results from the suggested algorithm will be measured. It is expected that while the suggested algorithm will generate a longer path, the dynamic constraints will be satisfied unlike in the wavefront solution.

Graph based planning offers a global approach. Graphs may be generated both deterministically (visibility graphs, cell decomposition, Voronoi diagrams), or randomly (probabilistic roadmaps). [10] Established algorithms (such as Dijkstra's shortest path search) are used to extract the shortest path from the graph [**?**]. However dynamics of the robot are ignored in the generation of the graph, and this is detrimental to the criterion of path smoothness. Challenges also arise when there are restrictive obstacles such as narrow passages, however research in areas such as probabilistic roadmaps (PRMs) offers promising solutions [**?**]. Optimization based planning offers more confidence in finding a global optimum, however it is hindered by a lack of robustness: a poorly formulated problem may not converge [**?**]. It is also difficult to define obstacles.

In an effort to provide a simple method of avoiding obstacles and defining a path, the manipulator's configuration space will be discretized using a grid-based approach, and curve fitting techniques will be employed to find the ideal path. Defining the environment with a grid-graph may not be the most effective (cell decomposition offers greater efficiency [**?**]), however it provides a simple way to defined obstacles as well as flexibility in representation. The grid will represent the configuration space for an articulated industrial robot. The grid cell size will define the computational resources required. A tradeoff between resolution in the grid and computational efficiency must be made.

To define the fitted curve, research has been done using Bzier curves [**?**]. Bzier curves provide smooth paths that guarantee obstacle avoidance. While very effective for lower-dimensional problems, quartic or quintic functions begin to get very computationally expensive. That is why this paper aims to use curve fitting in combination with genetic algorithms to optimize the curve. The curve will be optimized by minimizing distance, maximizing smoothness (minimizing jerk), and ensuring a safe passage around all obstacles. Genetic algorithms are ideal for such complex optimization problems and offer good robustness as well [**?**].

The father of genetic algorithms, John Holland, modeled this machine intelligence technique after the evolution research by Darwin and the genetic "survival of the fittest" discovered in the natural, biological sphere. There are two natural learning epitomes available: the brain and evolution. Holland was inspired by the fact that the processes of natural evolution and natural genetics have become elucidated over decades of study in biology and molecular biology [**?**]. The subtleties of the fundamental mechanisms of the brain, in contrast, are still shrouded in mystery. As such it seems clear to use the better understood model of genetic evolution as a platform for an optimization technique.

Research has already been done regarding the application of the parallel heuristic search method of genetic algorithms to curve fitting [**?**], [**?**], [**?**]. The goal is to find the coefficients of a polynomial function that will define a path that with optimized path criteria. Thus each chromosome will contain a set of function parameters, labelled $\{P_4, \ldots, P_0\}$ in Equation 1.

$$\phi = P_4 * \theta^4 + P_3 * \theta^3 + P_2 * \theta^2 + P_1 * \theta + P_0 \qquad (1)$$

Testing will be performed by creating random environments containing arbitrarily place objects. The algorithm will then plot a trajectory between a randomly defined start and end point. Many iterations of these tests will be performed in a variety of environments in order to accurately determine the effectiveness of this algorithm. The average path distance for each environment will be compared to the minimum distance as calculated by a Wavefront algorithm (CITE).

The wavefront algorithm is a deterministic algorithm which is capable of determining the shortest path between two points through a descritized environment. This will provide a "gold standard" which can be used to evaluate the performance of the genetical algorithm. Since the genetic algorithm is a stochastic search, it will never result in an identical path twice. As such all results will be presented as averages and their associated distributions.

The result will be a global path planning algorithm functional in a static environment. Dynamic environments are possible with the proposed algorithm, however computational efficiency must be optimized in order to minimize the time required to find the solution. This new applications offers a simple way to determine a path while taking into consideration all the path criteria.

III. Method

The Genetic Algorithm (GA) formed the basis for the experiemnts. Significant effort was expended setting up the GA in advance of testing, since without a well structured algorithm, the testing was sure to show poor results. The GA was implemented on Matlab, using the built-in GA libraries in order to avoid unnecessary development. While the base functionality was used, there remained many design considerations.

A. Criteria and Fitness

The aim of the implementation was to produce a path which minimized the obstacle collisions, length and jerk of the path represented by a polynomial function. The fitness of an individual was based on an equation considering these three criteria.

To consider obstacle collisions, the path was iterated along, and at each point along the path, a check was made to see if it was inside an obstacle. The number of units within an obstacle was penalized accordingly. Collisions resulted in high punishments to the fitness value of an individual. It

was important to move the path around a collision point, for a single collision in the final solution will render that path invalid.

Length was calculated based on the integral line length formula. The weight of the line length was set to a medium penalty. This was due to the logic that it is more important to find a valid solution that collides with no obstacles than finding the shortest path between the two points.

Finally, the maximum jerk value of the line was found by iterating the length of the fourth derivative of the line and recording the maximum value of this derivative. The jerk weight was set to make reduction in the jerk as important as reduction in the path length.

B. Configuration Space

Commanding a robotic manipulator's position is complicated by the fact that it cannot be specified is X-Y-Z coordinates, but rather must be commanded using joint angles. Not only is this not intuitive for a human, but existing planning techniques function using more standard X-Y or X-Y-Z coordinate systems. As such, the robot's workspace was converted into a Configuration Space. Figure 2 shows both the workspace and the configuration space for a 2-DOF robot in an arbitrary workspace. The robotic manipulator's two degrees of freedom are denoted by the blue and red lines (each DOF respectively). The alloable robot configurations denoted in the Configuration Space by whitespace, while collisions are denoted by black.

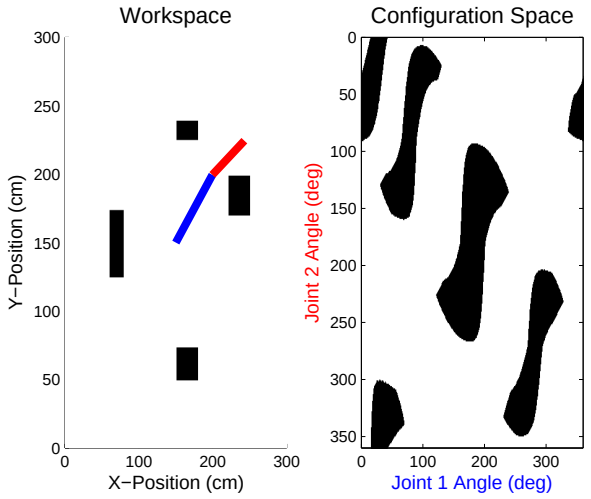

Fig. 1. Conversion from robot workspace to configuration space

In the configuration space, robot tracjectories can be easily identified by either a human or a motion planning algorithm. This angle-angle plot can then be used to implement a polynomial trajectory, of which optimal coefficients can easily be solved using a Genetic Algorithm. The parameterized trajectory is denoted in Equation 1.

C. Population Initiatlization

The population was randomly initialized with coefficient values in the range of 100 to 100. This range was imposed on the coefficients as it was found to provide a good compromise between promoting function manipulation while at the same time keeping it under control so that changes to coefficients did not make too drastic a change to the shape of the path.

The population size was chosen to be 75. This population number was found to provide the necissary diversity in the population. Lower population values were found to converge to non-ideal solutions.

D. Encoding

The GA was developed using a linear encoding scheme, where a chromosome was defined to be an array of the polynomial function's coefficients (denoted as $\{P_0, \ldots, P_3\}$ in Equation 1). Therefore, the GA would ultimately return the ideal set of coefficients for a function which would describe the optimal path for the robotic manipulator. These coefficients were limited to a range of [-500,500], since larger coefficients guarantee an invalid path since it would be too large and leave the workspace. While larger coefficients do not adversely affect the result of the algorithm, the increased diversity requires significantly larger computational resources which were not avaliable.

E. Linear Constraints

The path planning problem is a challenging optimization problem due to the various constraints involved. The goal is to generate a path that avoids all obstacles, as well as minimize the path length and minimize jerk. The problem is further constrained by the start and end points. The generated polynomial path must go through these desired start and end points.

To solve this constrained optimization problem, linear equality constraints are used. Linear equalities have the form:

$$A_{eq}x = b_{eq} \tag{2}$$

where x are the variables being optimized. For this project, the optimization variables are the coefficients of the polynomial.

For example, a second order polynomial $y = A + Bx + Cx^2$ through points $[12]$ and $[105]$ would have linear constraints of the form:

$$\begin{bmatrix} 1 & 10 & 10^2 \\ 1 & 5 & 5^2 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 10 \\ 5 \end{bmatrix} \tag{3}$$

F. Selection

A tournament selection method was used in this implementation. 2 individuals were chosen for each tournament. This allowed for an acceptable tradeoff between choosing a fit parent while promoting diversity in the offspring. This was one approach used to increase the diversity of the population when it was observed that the algorithm was getting stuck in local minima.

G. Reproduction and Mutation

A heuristic reproduction method was used wherein the child is generated by taking the mean of the coefficients of the two parents, and then biased to resemble the parent. In our implementation, the child was biased 20% towards the most fit parent using formula 4.

$$child_i = parent_{i2} + 1.2 * (parent_{i1} + parent_{i2}) \quad (4)$$

As is common in non-binary encoded GA problems, the crossover method used in this implementation is based on a mean of the parents' genes. This is a key distinction from a traditional point crossover method, as it allows the genes to evolve from values that were not present in the initial population. If point crossover methods had been used, the best case scenario would involve the optimal configuration of coefficients from the initial, random population.

Mutation was achieved applyinh a completely random mutation to a coefficient within the inidvidual selected to be mutated. In order to satisfy the linear constraints of the start and end points, if the function was rendered invalid by the mutation, a new random coefficient was chosen instead.

H. Termination

The termination criteria allow the GA to determine when it has reached the optimal solution to the problem. In this implementation, a termination criteria based on convergence of the fitness function was used. The mean fitness value over ten generations was averaged. When the this averaged mean fitness value converged within 0.001 units, the algorithm was terminated. This is representative of a situation where the diversity of the function has been reduced, and the algorithm has converged on a minimum fitness value.

I. Experiment

In order to assess the performance of the GA approach, an extensive experiment was devised. The algorithm was run 600 times on 15 sets of randomly generated points in 3 environments with randomly positionned obstacles. Invalid combinations of point sets were removed from the dataset and new points were randomly generated to replace them. The environments were chosen to represent environments which were sparsely populated with obstacles, through to dense population with obstacles. The results obtained by this experiment are presented in the following section.

IV. Results

As mentioned in section III-I, the following results were obtained through a test of 600 trials of running the genetic algorithm. Robot configuration spaces were generated based on physical environments which featured objects randomly placed in the robot's path. Three conficuration spaces were chosen based on how densly populated teh environment was with obstacles to present a range of environments in which to test the algorithm. Next, within each of these three environments, five sets of start and end points were generated randomly. The points were inspected to ensure that no invalid point configurations were present. The algorithm was then run 40 times on each point set, and with five point sets per environment and three environments, 600 trials were conducted.

Figure IV shows a visualization of 40 trials conducted on five startend point sets on a populated configuration space. Note that each line in the visualization represents a full evolution cycle with a population of 75 individuals.

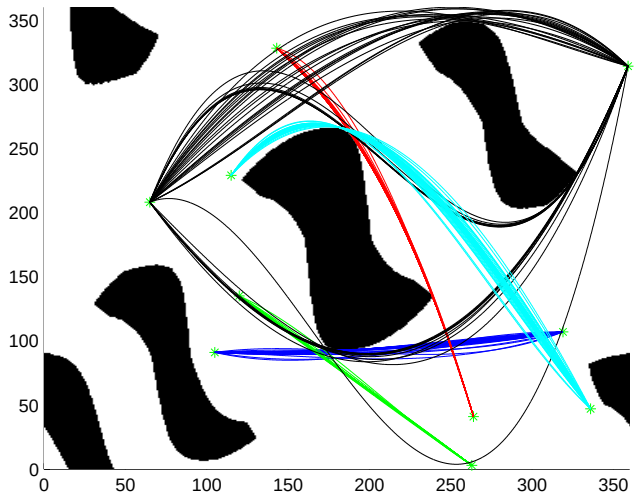

Fig. 2. Resultant optimized paths from 40 trials on five startend point sets. Axis represent the full range of motion of joints one and two of the manipulator.

The results of the experiment are presented in table **??**. number of trials run: 600

Criteria	Result	Unit	Note
Path length	0.7	% longer	Compared to wavefront method
Freq. of shorter path	79	%	Compared to wavefront method
Time to completion	5.15	seconds	20767% improvement on wavefront method
Average number of generations	12.3	generations	
Freq. of collision free path	94	%	
Jerk improvement	194	$\frac{D}{T^3}$	Compared to implementation not minimizing jerk.

TABLE I
RESULTS OBTAINED THROUGH 600 TRIALS OF THE PATH FINDING ALGORITHM.

V. Discussion

Evaluating the performance of the genetical algorithm (GA) was complicated by its ability to consider many factors

when determining the best path. These factors included path distance, vehicle dynamics and most importantly, obstacle avoidance. In order to determine a profile of the GA's performance, 600 tests were completed over the 15 random point configurations in 3 different environments.

The length of the path generated by the GA approach was on average 0.7% longer than the path generated by the wavefront method to which it was compared. The wavefront is a deterministic environment which will find a very short path around obstacles, with no consideration for vehicle dynamics. The GA works differently in that it is a stochastic algorithm that is sampling the environment in an effort to find the shortest path. As such, it can get stuck in a local minimum and not ultimately return the path of best fitness. While processes internal to the GA attempt to avoid these situations, it is not a perfect algorithm.

The GA was able to determine a shorter path than the wavefront algorithm 474 of 600 (79%) independent tests, despite attempting to also minimize jerk (sometimes at the cost of a longer path). While the GA is able to find a shorter path around obstacles 79% of the time, it is still on average 0.7% longer than wavefront since occasionally it will have an extrememly long path. These long paths are a result of the GA finding a local minimum and getting stuck there. In an actual implementation, the GA could be run multiple times in an effort to filter out local minima.

The GA was found to be much quicker at finding a solution than the wavefront algorithm. The wavefront algorithm took an average of 17 minutes and 50 seconds to analyze the shortest path between two points on any of the three test spaces used in this study. In comparison, the genetic algorithm took an average of 5.15 seconds to find the optimal path (20767% improvement). While the actual algorithm execution time is abitrary and based on algorithm implementation, three orders of magnitude faster

The algorithm was found to convege to a solution on average within 21.8 generations. This represents a quick convergence to the solution, and signaled that there was an effective diversity present in the algorithm.

If was expected to be able to develop an algorithm that would find a valid path $which contains no collisions$ 100% of the time. The results from the experiment show that the algorithm infact finds a collision-free path XX% of the time. This success rate is considered to be a success. In a real-life application, if the algorithm produces such an invalid path, it will be necissary to start the algorithm again from initialization.

A gold standard for the path with the shortest jerk could not be identified. Therefore, the metric by which the jerk of a path will be assessed will be to the results of the genetic algorithm when it is not trying to minimize jerk. 600 trials were performed with jerk minimization enabled. Another 600 trials were performed with the jerk criterion removed. It was found that the algorithm reduced jerk on average by XX%. Although this is not a large reduction in jerk, it does represent an improvement. The lower jerk path will always be chosen by a robot operator over a similar path with higher jerk.

VI. Conclusions

The results of the implementation of path planning using genetic algorithms presented in this report show that genetic algorithms can effectively plan a path through a series of obstacles, and present a opportunities to improve on existing path planning techniques. By selecting the appropriate fitness criteria, it is possible to bias your genetic search algorithm to find a path which satisfies multiple geometric conditions. In this implementation, the path was optimized for short length, no collisions and low jerk.

The algorithm is on average able to produce a path that is 100.7% the length of the shortest path as determined by the wavefront search method. 79% of the time, the algorithm can find a path that is shorter than that produced by the wavefront method. Collision avoidance is of critical importance in the application of path lanning to industrial manipulators. This implemenation of genetic algorithms could produce a collision free path 85% of the time. Finally, it was shown that the jerk of a path could be reduced by 11% when the fitness function included jerk criteria.

Through a manipulation from cartesian space to joint space to a configuration space, a method has been suggested to apply these path planning techniques to obstacle avoidance for industrial manipulators.

A main limitation of this algorithm is that the implementation using polynomials has a limited geography through which it is possible to evolve. Invalid solutions would be found by the algorithm whenever it was necessary for the path to double back, so that there are two y-axis values for one x-axis value. This is a fundamental limitation of a polynomial. In order to overcome this limitation, a function family would need to be selected that could double back across the same x-axis location.

References